1. Exchange's DAV Notification Implementation

Spec Title	Exchange's DAV Notification Implementation
Component	Server
Feature area	Future Protocols
Feature scope	Minor Feature - few dependencies
Related sections	<<related sections>>
Reference Sources	webdav-web/doc/ietf/draft-cohen-gena-p-base2.doc http://switchblade/draft-cohen-gena-p-base-01.doc
Product Version	Platinum
Author	Lisa Lippert
Feature Team	Zhidong Yang (dev) Randy Lehner (test)
Spec status	Reviewed by Development
Spec stability	Stable
Last Changed	1/20/1999

Deleted: 10/27/98

Deleted: 10/27/98

Revision Summary		
Date	Author	Changes
27/5/98	Lisadu	Initial Draft
10/6/98	Lisadu	Incorporated more information about the protocol and the Exchange store.
6/18/98	Lisadu	Added more detail from discussions with Zack, Brian and Joel.
7/15/1998	Lisadu	Added examples
8/5/1998	Lisadu	Clarifed what resource URIs are valid in a request -- because of bug 73462
8/14/1998	Lisadu	Changes from Alex's review -- more clarity, more scenarios.
10/5/1998	Lisadu	Updated for M3. Ready for wider review.
10/15/98	LisaL	Added notes (at end) on how notification will probably work with search
1027/98	LisaL	Changed model – ditched XML (see latest review summary)
11/3/1998	LisaL	Added detail for dealing with error cases I hadn't considered before
11/31/98	LisaL	Added detail on default depth values
12/2/1998	LisaL	Require 207 MULTISTATUS response on POLL all the time, rather than just some of the time, because it's easier for both clients and servers to handle.
1/12/1999	LisaL	Added detail on timeout limitations – section 1.7.7
1/20/1999	LisaL	Deleted section on M2 syntax – out-of-date and confusing.
5/12/1999	LisaL	RAID 105497: removed out-of-date requirement for subscription-id header to be returned
5/19/1999	LisaL	Described the support we now have (free from the store) for certain kinds of depth infinity notifications.
6/7/1999	LisaL	Fixed up various minor errors.
10/28/1999	LisaL	Depth "Infinity" notifications are cut.

Review Summary		
Date	Reviewer	General comments
8/14/1998	Alexhop	- Many areas need clarification. - XML needs proper namespaces. - Need more scenarios.
10/15/98	Brian Deen, Darren Shakib, Steve Wells, others	- Ditch the XML – this kind of detail is not required, since we have replication which provides detail (redundant) - Single timeout value for subscription (length of time between client refreshing the subscription)

Table of Contents

Deleted: 3
Deleted: 3
Deleted: 3
Deleted: 3
Deleted: 3
Deleted: 4
Deleted: 4
Deleted: 4
Deleted: 4
Deleted: 10/27/98
Deleted: 10/27/98

Deleted: 4
Deleted: 4
Deleted: 5
Deleted: 5
Deleted: 5
Deleted: 6
Deleted: 6
Deleted: 7
Deleted: 7
Deleted: 8
Deleted: 8
Deleted: 11
Deleted: 12
Deleted: 12
Deleted: 15
Deleted: 15
Deleted: 15
Deleted: 15
Deleted: 15
Deleted: 17
Deleted: 17
Deleted: 17
Deleted: 17

Deleted: 10/27/98
Deleted: 10/27/98

1.1 OVERVIEW

DAV Notifications will be implemented, based on Josh Cohen's internet-draft, to provide the following features for DAV/EX:

- Notify users of changes in the contents of lists (public or private folders)
- Notify users of new mail
- M3: Notify users of changes in individual messages

Note that this feature does not support the concept of a view: the notification engine does not send the client notifications when their view should be updated, but when the folder is updated and their view *may* be updated.

This work was completed in M3.

1.2 GOALS & OBJECTIVES

1.2.1 Requirements and Criteria

1.2.1.1 Performance/speed

Users should get notifications within 10 seconds of an event occurring. This is an arbitrarily chosen number to give the implementation some flexibility to allow high scalability at reasonable speed -- 10 seconds should be fairly reasonable.

1.2.2 Scenarios

1.2.2.1 "You have new mail"

The most important scenario for Exchange is to do new mail notifications well. Users must be able to subscribe to their own inbox: if anything changes in the inbox, the user is notified.

1.2.2.2 Other folder-based subscriptions

Users also must be able to subscribe to other folders, including both changes in their own folders and changes to public folders which they can view.

1.2.2.3 Message-based Subscription

These are the usage scenarios for being notified when an item changes within a folder, without being subscribed to the whole folder.

- A developer wants to be notified whenever a particular spec changes (the spec is stored in a public folder)
- An admin wants to be notified if details of a meeting change (location, invitees)

1.2.3 Features we must do

Support SUBSCRIBE, UNSUBSCRIBE and POLL methods.

Support NOTIFY method, using only call-back over UDP.

Expose store notification abilities.

Deleted: 10/27/98

Deleted: 10/27/98

1.2.4 Client Support

Currently client support for DAV is planned for Outlook 10. This is not scheduled to ship until after Platinum. Rosebud, Netdocs and VSS are also planning DAV support, though it is not yet known whether and how they will be using notification.

Outlook requirement: Must be able to display a view and update it when new items arrive, without downloading all the items in the folder to sort. Darren Shakib was going to follow up on this to figure out how it would be done, probably not using notifications.

This spec has now been reviewed by Steve Wells and others of the client team.

1.3 Open Issues

Resolve: Since the store requires a logon for the duration of a subscription, what logon are we using? Are we using a bunch or only one? One logon for every subscribed user gets us security…

1.3.1 Closed Issues

Resolve: can the store support depth=infinity notifications? No, not for any kind of subscription.

Deleted: .. Yes, for move, delete and update types of notifications, but not for newmember notifications.

1.4 Notification Features in Store

1.4.1 Notification triggers

The store supports a limited number of notification triggers. These are the ones we will expose through DAV:

- NewMail
- ObjectCreated
- ObjectDeleted
- ObjectModified (includes when properties of an object change)
- ObjectCopied
- ObjectMoved
- TableModified (properties of a table/folder change)

These are the notification triggers that we will not expose through DAV:

- CriticalError
- Reserved for MAPI
- Extended.
- StatusObject Modified

This notification trigger we may need, pending completion of a search spec:

- SearchComplete

1.4.2 Other Store features

- Can do notifications on individual messages changing
- Permissions: subscriptions are mapped to READ permisisons on an item
- Events are coalesced. The length of time that events are gathered before being sent is controlled by a single registry value, for all objects in the store and for all protocols. Default value for this is 1 second.

Deleted: <#>Can do depth=infinity notifications only for move, delete and update notifications (not for newmember)¶

Deleted: 10/27/98

Deleted: 10/27/98

- Store does both asynchronous and synchronous notifications. We will use the asynchronous notification method. This should meet speed requirements.
- Store requires a logon session as long as it is providing notifications. We will keep a logon object alive for the lifetime of the subscriptions.
- When the store gives us a notification, we will send it. There is no requirement in the protocol draft for notifications to occur at the beginning of an event, the end, or sometime shortly afterward, although latency should be small.

1.5 DAV NOTIFICATIONS PROTOCOL

The DAV Notification protocol is defined in http://search.ietf.org/internet-drafts/draft-cohen-gena-p-base-01.txt. This specification outlines a **snapshot** of Josh's spec and will not continue to change indefinitely as the internet-draft goes through the standards process.

This section provides a simple overview of GENA, even though we may not be supporting all features described in GENA, and lists the resolved and unresolved issues Exchange has or has had with GENA.

1.5.1 Headers

This is the list of headers defined by the GENA snapshot and whether we support them.

Header	Value	Description	Support?
Notification-type	*Type*[; subtype= *subtype]*	Type can be update, delete, move.	Yes
Call-back	URI	Address client wishes to be used for responses to the notification.	Yes
Subscription-lifetime	Seconds	Length of time before subscription times out.	Yes
Delivery-control	Poll-parameters	Valid poll-parameters are Wait-time (seconds), Poll-Interval (seconds), poll-provoke and batch-mode (can be "multipart/related").	No
Notifications-version	"http://extensions.iana.org/http/GENA/1/1"	Included on every request and response, to indicate the notifications protocol version for cross-compatibility	No
Subscription-ID	String	Near-unique identifier used to cross-reference	Yes

Deleted: create,

Resolved ISSUE: what is the difference between poll-control: poll-provoke and delivery-control: poll-provoke? Poll-control header was removed – so now poll-provoke is in the "delivery-control" header (it was originally in the "poll-control"

Wait-time is how long the client wants the server to keep the connection open. This is "persistent mode". We're not supporting it. If somebody specifies a wait-time, we'll respond back with

1.5.2 Methods

This is a list of methods defined in GENA. We support all of them.

1.5.2.1 SUBSCRIBE

Used by a client to subscribe to a folder or an item (property-level subscription is not supported).

1.5.2.2 UNSUBSCRIBE

Used by a client to unsubscribe to a subscription.

Deleted: 10/27/98

Deleted: 10/27/98

The server can cancel subscriptions if the client supports UDP callback. For example, if the server knows it is going to be unable to respond to subscriptions (e.g. the store drops out), it can cancel all subscriptions. Server can send an unsubscribe message to the client, with the cancelled subscription-Ids listed. It is not clear that this feature is needed by the client, so for now we will not support server-to-client UNSUBSCRIBE messages in Platinum.

Special case: If a client sends an unsubscribe with their callback-address specified instead of a subscription-ID (and with a resource URI of *), the server will remove all subscriptions to that callback-address. This would be nice from the point of view of a client, but would require a special index on the server, affecting performance, so we will not be implementing this in M2 or M3.

1.5.2.3 NOTIFY

The server sends NOTIFY packets over UDP to tell the client that something has changed. The subscription-ID indicates where the event occurred.

A NOTIFY method with no subscription-ID is an unsolicited notification. This can be used if the server wants to tell all clients, for example, that the store was shut down and notifications will not be working properly for a while. Optional – we will will not be supporting this in M2 or M3.

We will not be supporting use of NOTIFY over TCP.

1.5.2.4 POLL

POLL is used by the client:

- to ask the server whether there are any events in cases where the server is not sending notifications

1.5.3 Unresolved Protocol Issues

1.5.4 Resolved Protocol Issues

1.5.4.1 Use of "poll-provoke"

Issue: With the "poll-control" header created and the "poll-provoke" value, can we include a timeout value? I.e. Provoke poll for new information, before 30 seconds or we dump the information. Mail sent to sonuag and joshco to resolve this.

Resolution: we're no longer using poll-provoke.

1.5.4.2 Callback-interval

Resolved issue: We wanted a way for the client to specify how long the server should wait and collect notifications between callbacks. – Resolution: the client could always use POLL to control that.

1.5.4.3 Bundling Replies

The protocol needs to support bundling of notifications for responses to POLLs when a lot of events have piled up, and for multiple events that occur simultaneously or nearly so. This offers better performance.

Deleted: 10/27/98

Deleted: 10/27/98

Issues: How does a user POLL for all subscriptions? How does a user POLL for a subscription? Can a user POLL to some subscriptions and ask for callback on others?

Resolution: We can concatenate multiple notifications together, with separate methods and headers, when we need to.

1.5.4.4 Need more detail on response codes

Issues: what response code do we return if the subscription-ID and resource named in a request do not match up? "Subscription failed" should not be a 200-level code.

Resolution: response codes defined by Lisa, Sonu, Jesse & Josh. See below.

1.6 Implementation Decisions

This (below) is the final design and refers to M3.

1.6.1 Mapping store notification triggers to DAV notification sub-types

Many notifications fire on the folder parent as well as on the object itself.

When an object is copied from one folder to another, the GENA spec says that the event should show up as a "create" event in the target folder. However, we do not support the "create" type.

If the depth of the notification is not specified, it must be 0 for a non-collection resource, and it is assumed to be depth 1 for collections (default value).

What the client asks for:			What store event we get	Description of event
Ntype/subtype	Target	Depth		
Delete	*Any*	0	ObjectDeleted	The message or folder subscribed to was deleted.
	Folder	1	ObjectDeleted	A message or folder was deleted from the folder
Move	*Any*	0	ObjectMoved	The message or folder was moved.
	Folder	1	ObjectMoved	A message or folder was moved from or to the folder
Pragma/<http://schemas.microsoft.com/exchange/newmail>	Mailbox or folder	*Any*	NewMail	Special new mail update
	Message	*Any*	*None*	*Not valid – return 409 CONFLICT*
Update	Message	0	ObjectModified	The message was modified (either properties or body)
	Folder	0	TableModified	Properties of the folder were modified.
	Folder	1	ObjectCreated, ObjectModifed, ObjectDeleted, ObjectMoved, TableModified, ObjectCopied	A message or sub-folder was created in the folder, copied to the folder, moved to or from the folder, deleted from the folder, modified in the folder, or the folder properties were modified.
Update/newmember	*Any*	0	*None*	*Not valid – return 409 CONFLICT.*
	Existing Folder	1	ObjectCreated, ObjectMoved, ObjectCopied	A message or sub-folder was created in the folder, copied to the folder, or moved to the folder.
Update/propchange	*Any*	*Any*	*None*	*Not valid – return 409 CONFLICT. Exchange store does not distinguish between property changes on an item and changes in body.*
Any	Message	1	?	*Treat as depth = 0.*

Deleted: , inf

Deleted: , inf

Note that if a user asks for "update" notifications on a folder, depth=1, we need to get "TableModified", "ObjectCreated", "ObjectDeleted", "ObjectModified", "ObjectMoved" and "ObjectCopied" events.

Deleted: 10/27/98

Deleted: 10/27/98

1.6.2 Subscription-ID Header

A subscription-ID is a number assigned by the server to uniquely identify a subscription. The client uses this number to identify the subscription when using POLL and UNSUBSCRIBE. The server uses this number to identify the subscription when using NOTIFY and in response messages.

A client can request two subscriptions on the same resource for the same kind of event, and these will be the distinguishable only by subscription-ID. This is intended to make it easier for multiple clients on the same machine to subscribe to all the events they need independently. The subscription-ID is always included by the server so that the clients can sort out who gets what.

Multiple subscription-IDs can be specified in a SUBSCRIBE (renew), POLL or UNSUBSCRIBE request, as follows:

Subscription-ID: 1, 18, 243

These are all the cases in which subscription-ID header may or must be present:

SUBSCRIBE method: if the subscription-ID header is present and all the IDs match the content-location given, the server must try to renew the subscriptions.

200 Response to SUBSCRIBE: The successful response to a SUBSCRIBE method must include the subscription-Ids in the body, whether the user is trying to create a new subscription or renew one or more old subscriptions.

412 Response to SUBSCRIBE: If the client includes only bogus IDs in the Subscription-ID header when trying to renew a subscription, the server must fail the request. The failure response includes the subscription-ID header with every bogus ID included. E.g. if the client tries to renew 1234 and 518, and neither is valid, the server responds with 412 and with "Subscription-ID: 1234, 518" as a header.

Notet that if some are valid and some are invalid, the server must respond with a 207 Multi-status response (see below).

UNSUBSCRIBE method: The subscription-ID header must be present. If all IDs match the content-location, the server must cancel them all.

200 Response to UNSUBSCRIBE: The server must include the subscription-ID header with the ID's that were unsubscribed.

412 Response to UNSUBSCRIBE: As 412 response to UNSUBSCRIBE – the server lets the client know which subscription-IDs were bogus.

POLL method: The subscription-ID header must be present. All Subscription-IDs should match the same content-location.

200 Response to POLL actually in 207 Multistatus: Subscription-ID header is not present. The server returns a multistatus body with **ONLY** the subscription-IDs for subscriptions on which events happened. E.g. if the user polls on 518, 1234 and 2282, but events only happened on subscriptions 518 and 1234, then the server responds with 518 and 1234 in the body.

412 Response to POLL: As 412 response to SUBSCRIBE and UNSUBSCRIBE – the server lets the client know which subscription-IDs were bogus.

NOTIFY method: The subscription-ID header must be present. It includes the list of all subscription-IDs that are at the same content-location on which events happened.

1.6.3 How we implement the SUBSCRIBE Method

The SUBSCRIBE method MUST have either the notification-type header or the subscription-ID header.

Renewal of subscriptions: If the subscription-ID header is present, the SUBSCRIBE method is a subscription renewal, and the callback, notification-type, and depth headers SHOULD NOT be present. If

Deleted: 10/27/98

Deleted: 10/27/98

the subscription-ID header appears with the illegal headers, it is a bad request andmay be refused. (Note: See bug #118927 if you care just exactly how this works on the store).

1.6.3.1 Subscribing to folders

Exchange subscriptions are non-recursive. Subscriptions are not forwarded from receiving server to the server where the target object is. Client must subscribe to each folder separately on the server which hosts it.

1.6.3.2 Callback subscription example

NOTE: IN ALL OF THESE EXAMPLES, some headers have been left out for brevity, such as the Host: header. The Host: header in particular is needed for these requests to function.

C → S

```
SUBSCRIBE /mailbox/lisadu HTTP/1.1
Notification-type:
   pragma/<http://schemas.microsoft.com/exchange/newmail>
Call-back: httpu://myclient:88
Subscription-lifetime: 10000

S → C
HTTP/1.1 200 OK
Notification-type:
   pragma/<http://schemas.microsoft.com/exchange/newmail>
Call-back: httpu://myclient:88
Content-location: /mailbox/lisadu/
Subscription-lifetime: 5000
Subscription-ID: 1234
```

1.6.3.3 POLL subscription example

This is used when callback is impossible (e.g. firewalls between client and server). The client did not supply a callback address, therefore the client must send POLL requests to find out if something changed.

C → S

```
SUBSCRIBE /mailbox/lisadu/inbox/spam HTTP/1.1
Notification-type: update
Subscription-lifetime: 1000

S → C
HTTP/1.1 200 OK
Notification-type: update
Content-location: /mailbox/lisadu/
Subscription-lifetime: 5000
Subscription-ID: 1235
```

1.6.3.4 Subscription Renewal Example

C → S

```
SUBSCRIBE /mailbox/lisadu/inbox/spam HTTP/1.1
Subscription-ID: 1234
```

Deleted: 10/27/98

Deleted: 10/27/98

```
S → C
HTTP/1.1 200 OK
Subscription-ID:1234
Subscription-lifetime:5000
```

1.6.3.5 Notification-type Header

Must appear in new SUBSCRIBE request. Should not appear in SUBSCRIBE request if subscription-ID is specified to renew a subscription.

Must appear in SUBSCRIBE 200 OK response if anything in the notification-type or delivery control changed (may appear in all SUBSCRIBE 200 OK responses).

There can be no space in the notification-type. (not before nor after the '/' nor in the text)

BNF:

Notification-Type := "notification-type:" ntype

Ntype := "update" |

"update/newmember" |

"delete" |

"move" |

"copy" | //ISSUE – asked Josh for this

"pragma/<http://schemas.microsoft.com/exchange/newmail>"

1.6.3.6 Call-back Header

May appear in SUBSCRIBE request. If the call-back header is present and includes a call-back address, this means that the client has chosen the NOTIFY delivery-model.

If the call-back header is NOT present (or is empty), this means that the client has chosen a POLL delivery-model.

Contains a UDP URI.

BNF:

Call-back := "Call-Back:" URI

URI := "httpu://" *machine_name* ":" *port* "/" *path*

A UDP Callback URL is of the form: "httpu://*machine_name:port/path*" where the path may be defined by the client so that multiple clients can subscribe to the same events without conflict. Port must be used.

1.6.3.7 Subscription-lifetime Header

May be in SUBSCRIBE request. May be in SUBSCRIBE successful response (200 OK). If subscription-lifetime in the request is not accepted by the server, then the server must include the subscription-lifetime in the 200 OK response. If there is no subscription-lifetime in the request, it must be in the 200 OK response.

BNF:

Subscription-Lifetime := "Subscription-Lifetime:" 1*DIGIT

Deleted: 10/27/98

Deleted: 10/27/98

1.6.3.8 Content-location Header

It is left to the server implementation to decide how to choose the content-location header. The purpose of the content-location header is to allow the server to tell the client how to group subscriptions for best efficiency.

The Content-location header should be used by the client for all further requests concerning this subscription.

1.6.3.9 Response codes

200 OK	Subscription was successful. The server may have changed the parameters, such as poll-interval.
207 Multi-status	Multiple response codes to be found in XML body. Probably some of the subscription-IDs listed were invalid.
400 Bad Request	Probably an illegal combination of headers, or invalid notification type, or invalid combination of notification types. Depth: infinity requests get this error.
401 Unauthorized	User does not have access permissions or authorization to subscribe to this resource.
404 Not Found	Resource was not found.
406 Not Acceptable	SUBSCRIBE request had an accept-header that the server could not satisfy.
412 Precondition Failed	The subscription-ID(s) in the header did not match the resource named. This could be because the subscription-ID does not exist any more. Note that GENA draft now has this error code returned for unsupported notification type - we return actually 400 Bad Request.
415 Unsupported Media Type	Media type of body of SUBSCRIBE request not supported by the server.
501 Not Implemented	Server does not support the notification method (i.e. UDP callback, or delivery-control header).

Deleted: (not returned if the user asks for "create" type)

Deleted: 409 Conflict

Deleted: Invalid or unsupported notification-type.

For an example of the multi-status response, see the example given for POLL below. Valid subscription-IDs in the SUBSCRIBE request are included in the "200 OK" section of the multi-status response, and bogus subscription-IDs are included in the "412 Precondition Failed" section.

1.6.4 UNSUBSCRIBE

Required header: subscription-ID. Client must use the content-location given earlier by the server. Server may accept the original URL as well.

1.6.4.1 Example

C → S

```
UNSUBSCRIBE //mailbox/lisadu/ HTTP/1.1
Subscription-ID: 1234

S → C
HTTP/1.1 200 OK
Subscription-ID:1234
```

Deleted: 10/27/98

Deleted: 10/27/98

1.6.4.2 Response codes

200 OK	UNSUBSCRIBE was successful
207 Multi-status	Multiple status codes given. Probably some of the subscription-IDs given were invalid.
401 Unauthorized	User does not have access permissions or authorization to subscribe to this resource.
404 Not Found	Resource was not found.
412 Precondition Failed	One of the subscription-ID's in the header did not match the resource named. This could be because the subscription-ID does not exist any more. The subscription-IDs that did not match are included in a subscription-ID header in the response.

For an example of the multi-status response, see the example given for POLL below. Valid subscription-IDs in the UNSUBSCRIBE request are included in the "200 OK" section of the multi-status response, and bogus subscription-IDs are included in the "412 Precondition Failed" section.

1.6.5 NOTIFY

1.6.5.1 Call-back NOTIFY over UDP

A NOTIFY message is sent whenever events occur on a subscription, until the subscription times out. The NOTIFY packet will contain no data, because it is sent over UDP (insecure, and size restrictions) and because details can be gotten through replication. A subscription persists after a NOTIFY has been sent.

Note that in our implementation the subscription-ID header can be multi-valued. This is not consistent with GENA. It means that events fired on several notifications.

Call-back won't be precisely real-time because the store automatically coalesces events. MathruJ is working on synchronous notifications from the store, but Zack says we're not using that because synchronous notifications are necessary for some applications but not for us.

Because UDP NOTIFY packets may be lost, the server will keep track of past events on subscriptions. The server will continue to list all subscriptions for which events occurred, until the client sends some kind of acknowledgement (using the "subscription-ID" header in any method, especially POLL) that it is refreshing the subscription.

1.6.5.2 Example

```
S → C:  Event fires on 1234
NOTIFY httpu:myaddress HTTP/1.1
Subscription-ID: 1234

Client does not "refresh" the subscription, so later:

S → C:  Event fires on 1235
NOTIFY httpu:myaddress HTTP/1.1
Subscription-ID: 1234, 1235
```

1.6.5.3 Response Codes

200 OK	NOTIFY was successfully received.
401 Unauthorized	Server does not have permission to send notifications to this target. Server should terminate subscription.
404 Not Found	Subscription-ID was not found. Server should terminate subscription.

Deleted: 10/27/98

Deleted: 10/27/98

1.6.6 POLL

The URI used in the POLL method sent by the client MUST be the one provided in the content-location, and it MUST match the subscription-ID sent in the same method. This means that if multiple subscription-ID's are used, they must all be for the same content-location named in the URI.

The response to a POLL will contain the subscription-ID header with the list of changed subscriptions.

The server will consider the POLL message to be a "refresh" of all subscriptions identified by subscription-ID (meaning that the client is aware of an event on that subscription and the server doesn't have to continue sending that).

POLL can be used either with or without NOTIFY messages being sent from server to client – the server behaviour is the same. From the client perspective, POLL can be done over a firewall, and when NOTIFY call-back can't be used, is the only way of getting subscriptions in that case. Or, if NOTIFY callback can be and is being used, POLL is used by the client to refresh the subscription or to confirm (in case UDP packets were dropped) that nothing has happened on the subscription.

1.6.6.1 Error Messages for POLL

In some cases, since the client can poll multiple subscriptions, we MUST return a "multistatus" response code. The body of the multistatus response contains a status for each notification.

Since the client MUST be able to accept a multistatus response, we might as well use the multistatus response at all times. This is consistent with PROPPATCH and PROPFIND, and saves the client and the server from writing special-purpose code for the cases when all subscriptions have the same status, or for the case when there is only one subscription.

200 OK	Successful POLL. Events occurred since last POLL on subscriptions identified in headers. This response code only occurs within a 207 multi-status response body.
204 No Content	Successful POLL, but no events occurred. This response code only occurs within a 207 multi-status response body.
207 Multi-status	Status codes appear in the body for the various subscriptions that were polled.
401 Unauthorized	User does not have access permissions or authorization to POLL this resource. This response code only occurs within a 207 multi-status response body.
404 Not Found	Resource was not found. This response code only occurs within a 207 multi-status response body.
406 Not Acceptable	POLL contained an accept-header which could not be satisfied by server. This response code may be used by itself (not within a 207 multi-status response)
412 Precondition Failed	One of the subscription-ID's in the header did not match the resource named. This could be because the subscription-ID does not exist any more. The subscription-IDs that did not match are included in a subscription-ID header in the response.

1.6.6.2 Examples

This is a response to an "update" subscription on the user's mailbox, with our special "newmail" subtype. This means that all new mail generates an event, even if one of the new mails was routed to another folder by an inbox rule. The newmail subscription must be done on the user's mailbox. If inbox rules create copies of mail, all copies should be noted.

Deleted: 10/27/98

Deleted: 10/27/98

```
<D:multistatus xmlns:D  DA : >
  <D:response>
    <D:href>http://mailbox/lisal/</D:href>
    <D:status>HTTP/1.1 200 OK</D:status>
    <E:subscriptionID
   xmlns:E  http://schemas.microsoft.com/Exchange >
      <li>1</li>
      <li>2</li>
    </E:subscription-ID>
  </D:response>
  <D:response>
    <D:href>http://mailbox/lisal/</D:href>
    <D:status>HTTP/1.1 412 Precondition Failed </D:status>
    <E:subscriptionID>3</E:subscriptionID>
  </D:response>
</D:multistatus>
```

1.6.7 Keeping/Losing Subscriptions

Subscriptions will not be strongly persisted. This means that if the server goes down, the client's subscriptions will be lost, and the client will not know. The max-timeout applied by the server should be low (1 hour?) for all notifications, so that when the timeout is reached, correct state is restored by having the client either renew the subscription or at least be aware that it is ended. If we make the server so robust that it seldom loses notifications without canceling them, then the timeout can be high (1 day?). There MUST be a max timeout somewhere in that range. Note that there is a variation of +/- 5 minutes in server response to a timeout: e.g. if the client asks for a subscription with a timeout of 1 minute, it could be 6 minutes before the server actually times out the subscription.

Subscriptions will also be lost if:

- The resource is deleted (even if the resource is recreated).
- The resource is copied. Subscriptions will still exist on the original, not on the new copy.
- The resource is moved.

Subscriptions will not be cancelled by the server.

The client will return to a correct state by the time the subscription timeout is reached, or earlier if the client tries to do a POLL or UNSUBSCRIBE on the subscription.

1.6.8 Refreshing Subscriptions

If an event occurs, the client must acknowledge this in some way or the server will continue to notify the client of the same event. This is done with a POLL with subscription-ID header. The server responds – perhaps redundantly – with list of subscriptions on which events fired, then refreshes those subscriptions.

1.6.9 Timeout of Subscriptions

A subscription is timed out if it hasn't been refreshed (POLL) or renewed (SUBSCRIBE) for the entire period of the timeout.

Deleted: 10/27/98

Deleted: 10/27/98

1.6.10 Coalescing

The store coalesces events for a few seconds (according to its own logic) and them sends them all at once to the DAV engine. The DAV engine should concatenate the notifications together into one XML body to be sent as one response to the client when the client POLLs for that subscription.

1.6.11 Settings

Column1	Default Value	Administ rable?
Max-timeout	1 hour	Yes

1.7 Extended Example

The user subscribes to be notified of updates on a folder which is currently empty. The server allows the subscription.

```
C → S
SUBSCRIBE /private/foo/inbox/Empty%20Dir HTTP/1.1
Host: trumpet
Call-back: httpu://me
Notification-type: update

S → C
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 15 Jul 1998 18:49:24 GMT
Subscription-lifetime: 10000
Subscription-ID: 987
Content-location: /private/
Content-Length: 0

An event fires on subscription 987:

S → C
NOTIFY httpu://me HTTP/1.1
Subscription-ID: 987

The client doesn't respond, and an event fires on another
   ssubscription, 555.  Note that even though no new events fired on
   987, the server isn't sure that the client is aware of the first
   event, so the server includes 987 in the subscription-ID list:

S → C
NOTIFY httpu://me HTTP/1.1
Subscription-ID: 987, 555

The client did get the UDP notification, and sends the POLL to
   refresh the subscription.

C → S
POLL /private/ HTTP/1.1
Host: trumpet
Subscription-ID: 987, 555, 1228
```

Deleted: 10/27/98

Deleted: 10/27/98

```
In the server response, only the ID's for the subscriptions on which
   events fired are echoed back to the user:

S → C

HTTP/1.1 207 Multistatus
Server: Microsoft-IIS/5.0
Date: Wed, 15 Jul 1998 18:49:24 GMT
Content-Length: 0
Subscription-ID: 987, 555, 1228

<?xml version 1.0?>
<D:multistatus xmlns:D  DA : >
  <D:response>
    <D:href>http://mailbox/lisal/</D:href>
    <D:status>HTTP/1.1 200 OK</D:status>
   <E:subscriptionID
   xmlns:E  http://schemas.microsoft.com/Exchange >
      <li>987</li>
      <li>555</li>
    </E:subscription-ID>
  </D:response>
</D:multistatus>
</xml>

A new event fires on subscription 555 some time later…

S → C

NOTIFY httpu://me HTTP/1.1
Subscription-ID: 555

Etc.
```

1.8 NOTES

Instant Messaging is not supporting UDP.

Users need to have some kind of access right to objects they are subscribing to. The store maps the right to subscribe to the "read" right. This is not ideal for the long term.

1.8.1 Notification and Search

For dynamic searches: when the search suddenly gets a new match, this occurs by adding a link to the search results folder. Thus, the ordinary folder update event will do for dynamic search updates.

For static or dynamic searches that take a while to complete, the Search PM may define a new custom notification-type to handle this. This has not been spec'ed yet.

It will be the search client's responsibility to first start the search, learn where the search folder is, and then subscribe to know when the search updates or completes.

1.9 FEATURES WE'RE NOT DOING

1.9.1 Not in Platinum

Many of these should be considered in the future.

Deleted: 10/27/98

Deleted: 10/27/98

- List of new messages (otherwise client has to troll through many folders if a newmail notification arrives)
- Compounded requests
- Routing or proxying (respond with error 404 if client requests non-local resource)
- We will only support depth =1. Depth = infinity requests will be refused.
- We will not do IMPLICIT notification-type.
- Mailto callback mechanism
- TCP callback
- Notifications in the context of a client's view of a folder
- XML
- Delivery-control header
- Use of "subscription-ID" header in another method to refresh subscription
- Secure POLL (we don't verify the user's identity and permission to poll the subscription)
- Secure UNSUBSCRIBE (same)
- Special notification events for LOCK, UNLOCK, search complete. Right now the client must just ask repeatedly for the lockdiscovery property to see when lock status changes.

Compounded Requests

This is an explanation of what compounded requests are, and the issues which caused us to decide not to support them.

A compound request involves tacking the subscription onto a message like GET or SEARCH -- three extra headers are added. An implicit notification is the response to a compound request. Some issues with this feature: What happens if the GET request fails but the subscription succeeds, and vice versa? How does the server know if the respond to a SEARCH method would have changed? Is it constantly re-doing the search function? Our required features can be met more easily using a separate SUBSCRIBE method.

It will be possible in future versions to support simple compound requests, but to refuse complex ones. For example, if the user tries to do a GET an ASP file with a subscription tacked on, it is difficult to know when the results of a GET to that file would have changed, so the server can deny the request. If the user tries to do a GET on a DOC file, it is just the same as a SUBSCRIBE to the DOC file with a notification-type of "update", so this request can be handled. This is not a priority.

If a user wishes to do the GET and the subscribe at the same time to avoid the possibility of changes between the two commands, then pipelining can be used to achieve the same thing without all the problems of compounding.

In the meantime,with Platinum not supporting compounded or atomic requests, clients must do the SUBSCRIBE successfully **before** doing a GET on a resource to ensure that there are no changes that the client is unaware of.

Forwarding/Proxying

This implementation of DAV notifications will not support the forwarding of subscriptions. This section explains what proxying is and what the ramifications of not supporting proxying are.

The scenario for forwarding subscriptions is that the user wishes to subscribe to a remote object, but sends the subscription to their home DAV server. The DAV server subscribes directly to the remote object on behalf of the user, and forwards the notifications. This allows responses to subscriptions to be single-instanced to a DAV server even if that DAV server made the subscription on behalf of several users. However, we will not be implementing that feature.

Deleted: 10/27/98

Deleted: 10/27/98

Instead, if a client wishes to subscribe to changes on a public folder, the client must subscribe to a machine which hosts that public folder. If the client tries to subscribe to the wrong server, a 305 REDIRECT message will be sent so that the client then subscribes to the correct server.

Deleted: 10/27/98

Deleted: 10/27/98